Operating a Natural RPC Environment

This section mainly describes the tasks required to operate a Natural RPC environment. Some of these tasks are performed with the SYSRPC utility. For instructions on the functions the SYSRPC utility provides, refer to the Natural SYSRPC Utility documentation.

This section covers the following topics:

- Specifying RPC Server Addresses
- Stubs and Automatic RPC Execution
- Modifying RPC Profile Parameters Dynamically
- Executing Server Commands
- Logon to a Server Library
- Using the LOGON Option
- Using Natural RPC with Natural Security
- Using Natural RPC with EntireX Security
- Using Compression
- Using Secure Socket Layer
- Monitoring the Status of an RPC Session
- Handling Errors
- Terminating a Natural RPC Server

Specifying RPC Server Addresses

To each remote CALLNAT request, a server must be assigned (identified by *servername* and *nodename*) on which the CALLNAT is to be executed. Therefore, all subprograms to be accessed remotely must be defined:

- in a local service directory on the client side,
- or in a remote directory accessed via a remote directory server,
- or by way of default server addressing with the RPC profile parameter DFS,
- or within the client application itself by way of default server addressing.

In addition to the methods mentioned above, you can specify alternative servers.

If EntireX Broker is used, it is also possible to define servers using the EntireX location transparency.

Below is information on:

- Using Local Directory Entries
- Using Remote Directory Entries
- Specifying a Default Server Address at Natural Startup
- Specifying a Default Server Address within a Natural Session
- Using an Alternative Server
- Using EntireX Location Transparency

Using Local Directory Entries

All data of a client’s local service directory is stored in the subprogram NATCLTGS. At execution time, this subprogram is used to retrieve the target server. As a consequence, NATCLTGS must be available in the client application or in one of the Natural steplibs defined for the application.

If NATCLTGS has not been generated into a steplib or resides on another machine, use the appropriate Natural utility (SYSMAIN, SYSTRANS or the Natural Object Handler) to move NATCLTGS into one of the steplib defined for the application.

If you are using a NATCLTGS for joint usage, you must make it available to all client environments, for example by copying it to the library SYSTEM, or, if an individual copy is used for a client, it must be maintained for this client using the Service Directory maintenance function of the SYSRPC utility.

To define and edit RPC service entries, see the section Service Directory Maintenance in the SYSRPC Utility documentation.

Using Remote Directory Entries

A remote directory contains service entries that can be made available to several Natural clients. The Natural clients can retrieve these service entries from remote directory servers. For information on the purpose and on the installation of remote directory servers, see Using a Remote Directory Server.

For information on the SYSRPC Remote Directory Maintenance function, see the relevant section in the SYSRPC Utility documentation.

Specifying a Default Server Address at Natural Startup

Instead of addressing a server by using a local or remote service directory, you can preset a default server with the RPC profile parameter DFS, as described in your Natural Operations documentation. This server address is used if the subprogram can be found in neither the local nor the remote service directory.

The DFS setting determines the default server for the whole session or until it is overwritten dynamically.

If no DFS setting exists and the server address of a given remote procedure call could not be found in the service directory, a Natural error message is returned.

A default server address defined within a client application remains active even if you log on to another library or if a Natural error occurs.

Specifying a Default Server Address within a Natural Session

The client application itself may dynamically specify a default server address at runtime. For this purpose, Natural provides the application programming interface **USR2007N** in the library SYSEXT. The interface enables you to determine a default server address that is to be used each time a remote program cannot be addressed via the service directory. It includes the following parameters:

Parameter	Format	Explanation
function	**A1**	**P** Put: Determines that the server address (composed of the parameters *nodename* and *servername*, see below) is the default address for all subsequent remote procedure calls not defined in the service directory. To remove a default server address, specify a "blank" for *nodename* and *servername*. **G** Get: Retrieves the current default server address as set by the function P.
nodename	**A192**	Specifies/returns the name of the server node to be addressed. The node name may have up to 32 characters for physical node names and up to 192 characters for logical node names. See Using EntireX Location Transparency. **Note:** For compatibility reasons, *servername* is defined with BY VALUE RESULT to support existing callers which pass an A8 field for the *servername*.
servername	**A192**	Specifies/returns the server name to be addressed. The server name may have up to 32 characters for physical server names and up to 192 characters for logical service names. See Using EntireX Location Transparency. **Note:** For compatibility reasons, *nodename* is defined with BY VALUE RESULT to support existing callers which pass an A8 field for the *nodename*.
logon	**A1**	Specifies/returns the logon option.
protocol	**A1**	Specifies/returns the transport method. Valid value: B (=EntireX Broker).

The Natural subprogram NATCLTPS in the library SYSRPC is only maintained for compatibility reasons.

Using an Alternative Server

To avoid connection failures, you may want to define several alternative servers for a remote CALLNAT. If you specify such alternative servers, Natural proceeds as follows:

- The client makes a first attempt to establish the connection.
- If this attempt fails, instead of providing an error message, a second attempt is made, however, this time not on the same server. Instead, the service directory is searched again starting at the current entry to find out whether or not another server is available which offers the desired service.
- If a second entry is found, Natural tries to establish the connection to this server. If the remote procedure call is performed successfully, the client application keeps on running. The user does not notice whether the connection to the first server or to the alternative server produced the result.
- If no further entry is found or if the connection to alternative servers fail, Natural issues a corresponding error message.

To enable the use of an alternative server

1. Define more than one server in the service directory for the same service.
2. Set the Natural RPC profile parameter TRYALT to ON to give permission to use an alternative server.

This parameter can also be set dynamically for the current session See the Parameter Maintenance function as described in the SYSRPC Utility documentation.

Using EntireX Location Transparency

Using EntireX location transparency, you can change physical node and server names without having to configure anything or to change client and/or server programs. Now, instead of using a physical node and physical server name, a server can be addressed by a logical name. The logical name is mapped to the physical node and server names using directory services.

To take advantage of location transparency, the Natural RPC has been enabled to accept a logical name wherever only a node and server name could be specified before. The logical name is passed to the EntireX Broker before it is used the first time.

The maximum length of a logical name is 192 characters. To avoid new Natural profile parameters, a logical name is specified in the server name part of the already existing parameters. There are two kinds of logical names:

- **Logical node names**
  With a logical node name you specify a logical name for the node only in conjunction with a real server name. A logical node name can be used in all places where you can also use a real node name. To define a logical node name the keyword LOGBROK must be used.
  Example:
  `SRVNVODE='LOGBROK=`*`logical_node_name,my_set'`*
- **Logical services**
  With a logical service, you specify a logical name for both the node and the server. A logical service can be used in all places where you can also use a real node and server name. To define a logical service, the node name must be set to * (intentionally left empty), and the server name contains the logical service name.
  Example:
  `SRVNVODE='*' SRVNAME='`*`logical_service_name,my_set'`*
  Note:
  In the case of interface USR2071N, you can LOGON to a logical service name by using the keyword LOGSERVICE together with the logical service name in the field *broker-id*.

For more details about the EntireX location transparency, refer to the EntireX documentation.

The following components refer to node and server names:

- Natural profile parameters SRVNODE, SRVNAME, DFS and RDS
- Service maintenance of the SYSRPC utility
- Service directory (NATCLTGS)
- User application interfaces USR2007N, USR2071N
- Service programs RPCERR, RPCINFO

See also Location Transparency in Service Directory Maintenance in the Natural SYSRPC utility documentation.

Stubs and Automatic RPC Execution

Stubs are no longer required if automatic Natural RPC execution is used, as described in Working with Automatic Natural RPC Execution below.

However, generating stubs provides the advantage of controlling the CALLNAT(s) executed remotely and facilitates error diagnoses. Should a remote call fail due to an incorrect CALLNAT name, the Natural error message issued then helps to immediately identify the problem cause. Without a stub, for an incorrect CALLNAT you may receive follow-up errors returned from the transport layer or the Natural server.

Below is information on:

- Creating Stub Subprograms
- Working with Automatic Natural RPC Execution

Creating Stub Subprograms

With the Stub Generation function of the SYSRPC utility, you can generate the Natural stub subprograms used to connect the client’s calling program to a subprogram on a server. The stub consists of a parameter data area (PDA) and of the server call logic.

The PDA contains the same parameters as used in the CALLNAT statement of the calling program and must be defined in the Stub Generation screen of the Stub Generation function. If a compiled Natural subprogram with the same name already exists, the PDA used by this subprogram is used to preset the screen. The server call logic is generated automatically by the Stub Generation function after the PDA has been defined.

At execution time, the Natural application program containing the CALLNAT statement and the stub subprogram must exist on the client side. The Natural application subprogram must exist on the server side. Both the stub and server subprograms must have the same name.

For information on the SYSRPC Stub Generation function, see the relevant section in the SYSRPC Utility documentation.

Working with Automatic Natural RPC Execution

You are not required to generate Natural RPC stubs, but you can work with automatic Natural RPC execution (i.e. without using Natural stubs). To work with automatic Natural RPC execution set the RPC parameter AUTORPC as follows:

```
AUTORPC=ON
```

In that case, you can omit the generation of the client stub during your preparations for RPC usage. When the automatic Natural RPC execution is ON, Natural behaves as follows:

- if a subprogram cannot be found locally, Natural tries to execute it remotely (a stub subprogram is not needed),
- the parameter data area will then be generated dynamically during runtime.

As stubs only exist for client programs, this feature has no effect on the CALLNAT program on the server.

If AUTORPC is set to ON, and a Natural stub exists, it will still be used.

Modifying RPC Profile Parameters Dynamically

With the Parameter Maintenance function, you can dynamically modify some of the RPC profile parameters set in the Natural profile parameter module for the current session.

Attention:
These modifications are retained as long as the user session is active; they are lost when the session is terminated. Static settings are only made using Natural profile parameters.

Executing Server Commands

Active servers that have been defined in the service directory (see Specifying RPC Server Addresses) can be controlled with the SYSRPC server command execution function as described in the relevant section in the SYSRPC Utility documentation.

Logon to a Server Library

The server library on which the callnat is executed depends on the RPC LOGON option on the client side and a couple of parameters on the server side.

The following table shows which the relevant parameters are and how they influence the library setting:

	Client		Server				
	1	2	3	4	5	6	7
	*library-id	RPC LOGON flag for server entry set?	LOGONRQ set?	Server started with STACK=	NSC or native Natural?	NSC: RPC LOGON option in library profile	Server *library-id
1	Lib1	no	no	logon lib1	No influence	N/--	Lib1
2	Lib1	no	no	logon lib2	No influence	N/--	Lib2
3	Lib1	no	yes	(Client LOGON flag = no) **and** (LOGONRQ = yes) is not possible.			
4	Lib1	yes	No influence	No influence	NSC	AUTO	Lib1
5	Lib1	yes	No influence	No influence	NSC	N	Lib1
6	Lib1	yes	No influence	No influence	Native Natural	--	Lib1

Explanation of the table columns:

1. The library ID of the client application where the callnat is initiated.
2. The value of the RPC LOGON flag. Can be set for a whole node or a server.
   The flag can be set by using
   the Service Directory Maintenance function of the SYSRPC utility,
   or the DFS parameter,
   or the application programming interface USR2007N.
3. LOGONRQ can be set as a Natural profile parameter at server startup.
4. The library ID to which the server is positioned at its startup.
5. Does the server run under Natural Security (NSC) or not?
6. The setting of the LOGON option in the NSC library profile (Session options > RPC restrictions) of the NSC server application. If the NSC LOGON option is set to AUTO, only library and user ID are taken. If set to N (default), the library, user ID and password parameters are evaluated.
7. The library on the server where the CALLNAT program is finally executed.

Using the LOGON Option

The LOGON option defines on which library the remote subprogram is to be executed. See also Logon to a Server Library.

When you do not use the LOGON option, the CALLNAT is executed on the library to which the server is currently logged on. This server logon is defined with the Natural profile parameter STACK = (LOGON *library*). The server will search for the CALLNATs to be executed in *library* (and all associated steplibs defined for *library*).

A client application can be enabled to execute a subprogram on a different library by setting the LOGON option for this subprogram. This causes the client to pass the name of its current library to the server, together with this LOGON option. The server will then logon to this library, searching it for the desired subprogram and, if the latter is found, it will execute it. After that, it will make a logoff to the previous library.

Settings Required on the Client Side

To set the LOGON option, you can use either the SYSRPC Service Directory maintenance function (see the relevant section in the SYSRPC Utility documentation) or - when using a default server - the DFS profile parameter or the Interface USR2007N.

Settings Required on the Server Side

No setting is required on the server side.

Using Natural RPC with Natural Security

Natural RPC also supports Natural Security in client/server environments, where security may be active on either (or both) sides. If security data is to be passed to the server, the LOGON option (see also Using the LOGON Option) must be used.

The user ID and password are established as follows:

- **If the client runs under Natural Security:**
  The user ID and password from the Natural Security logon on the client are used and passed to the server.
- **For non-Natural Security clients:**
  The application programming interface **USR1071N** is provided which the user has to call for specifying logon data which are then passed to the server. USR1071N is contained in the library SYSEXT. The logon data contains the user ID and password from which the so-called security token is generated, and additionally some administrative information.
  Two samples are provided: **USR1071P** which is passing just user ID and password, and **USR1071X** (extended version) which enables the user in addition to set/retrieve various data.
  To invoke, for example, USR1071P from within a program, specify:
  `FETCH RETURN 'USR1071P' USERID(A8) PASSWORD(A8).`
  For a more detailed description, see the **USR1071T** member in library SYSEXT.

If the server runs under Natural Security, the user ID and password from the client are verified against the corresponding user security profile on the server, and the logon to the requested library and the execution of the subprogram are performed according to the corresponding Natural Security library and user profile definitions on the server.

After the execution of the subprogram, the library used before the CALLNAT request is made current again on the server. In the case of a conversational RPC, the first CALLNAT request within the conversation sets the library ID on the server; and the CLOSE CONVERSATION statement resets the library ID on the server to the one before the conversation was opened.

To enforce the LOGON option - that is, if you want a server to accept only requests from clients where the LOGON option is set - set the profile parameter/subparameter LOGONRQ to ON for the server.

As part of the Natural RPC Restrictions in library profiles of Natural Security, a session option "Close all databases" is provided. It causes all databases which have been opened by remote subprograms contained in the library to be closed when a Natural logon/logoff to/from the libraries is performed. This means that each client uses its own database session. See Natural RPC Restrictions in the Natural Security documentation.

Using Natural RPC with EntireX Security

Natural RPC fully supports EntireX Security on the client side and the server side.

Client Side

To logon to and logoff from the EntireX Broker, the application programming interface USR2071N is provided in library SYSEXT. To logon to EntireX Broker, you use the logon function of USR2071N and pass your user ID and password to the selected EntireX Broker. After a successful logon, the security token returned is saved by Natural and passed to the EntireX Broker on each subsequent call. The logon feature is fully transparent to the Natural application.

If EntireX Security has been installed or if AUTOLOGON=NO has been specified in the EntireX Broker attribute file, you must invoke USR2071N with the logon function before the very first remote CALLNAT execution.

You are recommended to invoke USR2071N with the logoff function as soon as you no longer intend to use a remote CALLNAT.

Using the Application Programming Interface USR2071N

USR2071N has the following parameters:

Parameter	I/O	Format	Description
function	I	A08	Function code. Values: LOGON Logon to EntireX Broker LOGOFF Logoff from EntireX Broker
broker-id	I	A192	Broker ID The *broker-id* may have up to 32 characters for physical node names and up to 192 characters for logical node names or logical service names. See Using EntireX Location Transparency. **Note:** For compatibility reasons *broker-id* is defined with BY VALUE RESULT to support existing callers which pass an A8 field for the *broker-id*.
user-id	I	A08	User ID.
password	I	A08	User ID’s password.
newpassw	I	A08	User ID’s new password.
rc	O	N04	Return value: 0 ok 1 invalid function code 9999 EntireX Broker error (see *message*)
message	O	A80	Message text, returned by EntireX Broker.

The Subprogram USR2071N should be copied to the Library SYSTEM or to the steplib library, or to any application.

The parameters listed above must be defined via DEFINE DATA.

The calling program must contain the following statement:

```
CALLNAT 'USR2071N' FUNCTION BROKER-ID USER-ID PASSWORD NEWPASSW                   RC MESSAGE
```

Special considerations when using location transparency:

- If you want to LOGON using a logical node name, you have to use the LOGBROK keyword.

  ```
  BROKER-ID := 'LOGBROK=my_logical_node,my_set'
  ```

- If you want to LOGON using a logical service name, you have to use the LOGSERVICE keyword.

  ```
  BROKER-ID := 'LOGSERVICE=my_logical_service,my_set'
  ```

Functionality

LOGON

An EntireX Broker LOGON function is executed to the named *broker-id* with the *user-id* and the *password* passed. After a successful LOGON call, the client can communicate with the EntireX Broker *broker-id* as usual.

With *newpassw* the client user can change her/his password via the EntireX Security features.

Notes:

- If a successful LOGON has been performed, the user ID used in this LOGON will be passed to the named EntireX Broker on all subsequent remote procedure CALLNATs which are routed via this EntireX Broker. Without an explicit LOGON, the current contents of *USER is used. The same applies if you have issued a LOGON to EntireX Broker 1, but your remote procedure CALLNAT is routed via EntireX Broker 2.
- It is possible to concurrently LOGON to multiple EntireX Brokers. For each LOGON, a different user ID may be used.
- The user ID used for the LOGON to the EntireX Broker may be different from the Natural user ID under which the client application runs.
- An internal reLOGON is done after an EntireX Broker timeout has occurred, if the original LOGON was done without a password (the password used in the LOGON is not saved). If no internal reLOGON is possible after a timeout has occurred, the client has to explicitly reissue the LOGON.
- At the end of the Natural session, an implicit LOGOFF is executed to all EntireX Brokers to which a LOGON has been performed.

LOGOFF

An EntireX Broker LOGOFF function is executed to the *broker-id* named.

Server Side

If the value of ACIVERS is 2 or higher, the server will log on to the EntireX Broker at the session start using the LOGON function. The user ID is the same as the user ID defined by SRVUSER.

If EntireX Security has been installed and if the EntireX trusted user ID feature is not available, there are two alternative ways to specify the required password:

- **SRVUSER=*NSC**

  If Natural Security is installed on the server, you can specify SRVUSER=*NSC to determine that the current Natural Security userID which was used when the server was started is used for the LOGON in conjunction with the accompanying Natural Security password. In this case, the value set for ACIVERS must be at least 4.

- **USR2072N**

  The application programming interface USR2072N enables you to specify a password which is used for the LOGON in conjunction with SRVUSER.

Using the Interface USR2072N

USR2072N has the following parameter:

Parameter	I/O	Format	Description
password	I	A08	User ID’s password.

The Subprogram USR2071N should be copied to the library SYSTEM or to the steplib library, or to any application.

The parameter listed above must be defined using the DEFINE DATA statement.

The calling program must contain the following statement:

```
CALLNAT 'USR2072N' PASSWORD
```

The calling program must be executed before the Natural RPC server has started its initialization. To accomplish this, put the name of the calling program on the Natural stack when starting the server:

```
STACK=(LOGON server-library;USR2072P password)
```

Using Compression

Compression types may be: 0, 1 or 2. Stubs generated with COMPR = 1 or 2 can help reduce the data transfer rate.

Compression Type	Description
COMPR=0	All CALLNAT parameter values are sent to and returned from the server, i.e. no compression is performed.
COMPR=1 (default)	M-type parameters are sent to and returned from the server, whereas O-type parameters are only transferred in the send buffer. A-type parameters are only included in the reply buffer. The reply buffer does not contain the Format description.
COMPR=2	Same as for COMPR = 1, except that the server reply message still contains the format description of the CALLNAT parameters. This might be useful if you want to use certain options for data conversion in the Software AG product EntireX Broker (for more information, see the description of Translation Services in the EntireX Broker documentation).

Using Secure Socket Layer

The Natural RPC supports Secure Socket Layer (SSL) for the TCP/IP communication to the EntireX Broker.

To enable the EntireX Broker to recognize that the TCP/IP communication should use SSL, you must use one of the following methods:

- Append the string :SSL to the node name. If the node name has already been postfixed by the string :TCP, :TCP must be replaced by :SSL.
- Prefix the node name with the string //SSL:

Example:

```
SRVNODE=’157.189.160.95:1971:SSL’
```

Before you access an EntireX Broker using SSL, you must first invoke the application programming interface USR2035N to set the required SSL parameter string

Using Interface USR2035N

USR2035N has the following parameters:

Parameter	I/O	Format	Description	
function	I	A01	Function code. Values:	
			P	Put: Specify a new SSL parameter string.
			G	Get: Retrieve previously specified SSL parameter string.
SSLPARMS	I	A128	SSL parameter string as required by the EntireX Broker	

The Subprogram USR2035N should be copied to the library SYSTEM or to the steplib library, or to any application.

The parameters listed above must be defined via DEFINE DATA.

The calling program must contain the following statement:

```
CALLNAT ’USR2035N’ FUNCTION SSLPARMS
```

Functionality of Interface USR2035N

P (specify a new SSL parameter string)

The SSL parameter string is internally saved and passed to EntireX each time an EntireX Broker using SSL communication is referenced the first time. You may use different SSL parameter strings for several EntireX Broker connections by calling USR2035N each time before you access the EntireX Broker the first time.

Example:

```
FUNCTION := ’P’
SSLPARMS := ’TRUST_STORE=FILE://DDN:CACERT&VERIFY_SERVER=N’
CALLNAT ’USR2035N’ USING FUNCTION SSLPARMS
```

To set SSL parameters in case of an Natural RPC server, put the name of the calling program onto the Natural stack when starting the server.

Example:

```
STACK=(LOGON server-library;set-SSL-parms)
```

Where *set-SSL-parms* is a Natural program that invokes the user application programming interface USR2035N to set the SSL parameter string.

G (retrieve previously specified SSL parameter string)

The previously put SSL parameter string is returned to the caller.

For more information about the SSL parameter string, refer to the EntireX documentation.

Monitoring the Status of an RPC Session

This part is organized in the following sections:

- Using the RPCERR Program
- Using the RPCINFO Subprogram
- Using the Server Trace Facility
- Defining the Trace File

Using the RPCERR Program

You can use the RPCERR program from the command line or invoke it via FETCH from within a Natural program.

RPCERR displays the last Natural error number and message if it was RPC related and it also displays the last BROKER reason code and associated message. Additionally, the node and server name from the last EntireX Broker call can be retrieved.

Example of an RPC Error Display: RPCERROR

```
 NATURAL error number: NAT6972
   NATURAL error text :
   Directory error on Client, reason 3 :3:.

 RPC error information:
   No additional information available.

 Server Node: Library:      SYSRPC
   Server Name:      Program: NATCLT3
                              Line No: 0480
```

Using the RPCINFO Subprogram

You can use the subprogram RPCINFO in your application program to retrieve information on the state of the current RPC session. This also enables you to handle errors more appropriately by reacting to a specific error class.

The subprogram RPCINFO is included in the library SYSRPC.

Example:

```
DEFINE DATA LOCAL USING RPCINFOL
  LOCAL
  1 PARM   (A1)
  1 TEXT   (A80)
  1 REDEFINE TEXT
    2 CLASS  (A4)
    2 REASON (A4)
END-DEFINE
...
OPEN CONVERSATION USING SUBPROGRAM ’APPLSUB1’
   CALLNAT ’APPLSUB1’ PARM
CLOSE CONVERSATION *CONVID
...
ON ERROR
   CALLNAT ’RPCINFO’ SERVER-PARMS CLIENT-PARMS
   ASSIGN TEXT=C-ERROR-TEXT
   DISPLAY CLASS REASON
END-ERROR
...
END
```

RPCINFO has the following parameters which are provided in the PDA RPCINFOL:

Parameter	**Format**	**Description**
SERVER-PARMS		Contains information about the Natural session when acting as a server. The SERVER-PARMS only apply if you execute RPCINFO remotely on an RPC server.
S-BIKE	A1	Transport protocol used. Possible value: **B** (EntireX Broker)
S-NODE	A8	The node name of the server.
S-NAME	A8	The name of the server.
S-ERROR-TEXT	A80	Contains the message text returned from the transport layer.
S-CON-ID	I4	Current conversation ID. Note that this is the physical ID from EntireX Broker, not the logical Natural ID. This parameter always contains a value as EntireX Broker generates IDs for both conversational and non-conversational calls. If the physical conversation ID is either non-numeric or greater than I4, a -1 is returned.
S-CON-OPEN	L	Indicates whether there is an open conversation. This parameter contains value TRUE if a conversation is open, otherwise it contains FALSE.
CLIENT-PARMS		Contain information about the Natural session when acting as a client. The CLIENT-PARMS only apply if you execute RPCINFO remotely on an RPC client.
C-BIKE	A1	Transport protocol used. Possible value: **B** (EntireX Broker)
C-NODE	A8	The node name of the previously addressed server.
C-NAME	A8	The name of the previously addressed server.
C-ERROR-TEXT	A80	Contains the message text returned from the transport layer.
C-CON-ID	I4	Conversation ID of the last server call. Note that this is the physical ID from EntireX Broker, not the logical Natural ID. If no conversation is open, the value of this parameter is less than or equal to 0. If the physical conversation ID is either non-numeric or greater than I4, a -1 is returned.
C-CON-OPEN	L	Indicates whether there is an open conversation. This parameter contains value TRUE if a conversation is open, otherwise it contains FALSE.

Using the Server Trace Facility

Natural RPC includes a trace facility that enables you to monitor server activities and trace possible error situations.

Activating/Deactivating the Server Trace Facility

To activate/deactivate the server trace facility, start the server with the option

`TRACE=`*`n`*

The integer value "*n*" represents the desired trace level; that is, the level of detail in which you want your server to be traced. The following values are possible:

Value	Trace Level
0	No trace is performed (default).
1	All client requests and corresponding server responses are traced and documented.
2	All client requests and corresponding server responses are traced and documented; in addition, all RPC data are written to the trace file.

The RPC trace facility writes the trace data to the Natural Report Number 10.

Defining the Trace File

The trace file definition depends on the environment. The following topics are covered below:

- Trace File Handling for Mainframe Environments - General Information
- Trace File Handling in OS/390 Batch Mode
- Trace File Handling under CICS
- Trace File Handling in VSE/ESA Batch Mode
- Trace File Handling in BS2000/OSD Batch Mode

Trace File Handling for Mainframe Environments - General Information

On the mainframe, define the trace file appropriate to your environment, see also the NTPRINT macro in the Natural Operations Manual for Mainframes.

Trace File Handling in OS/390 Batch Mode

a) Running A Server As Single Task

In the server start job, assign an OS/390 dataset to the Natural additional report CMPRT10.

Example:

```
//NAT31    JOB  CLASS=K,MSGCLASS=X
//NATSTEP  EXEC PGM=NATOS31
//STEPLIB  DD   DISP=SHR,DSN=SAG.NAT.LOAD
//         DD   DISP=SHR,DSN=SAG.ETB.LOAD
//CMPRMIN  DD   *
IM=D,MADIO=0,MT=0,OBJIN=R,AUTO=OFF,MAXCL=0,ID=’,’,INTENS=1,
PRINT=((10),AM=STD)
/*
//SYSUDUMP DD   SYSOUT=X
//CMPRT10  DD   SYSOUT=X
//CMPRINT  DD   SYSOUT=X
/*
```

b) Running a Server With Replicates

1. Set the RPC parameter NTASKS to a value greater than 1.
2. Assign CMPRMIN to a dataset with DISP = SHR or to *.
3. As each task writes on a separate CMPRINT dataset, define the following DD card names:
   CMPRINT for the main task;
   CMPRINT1 to CMPRINT9 for the first nine subtasks;
   CMPRIN10 to CMPRIN*nn* for the next two-digit numbers of subtask, *nn* = NTASKS-1.
4. If the RPC subparameter TRACE is set, the trace facility writes to Printer 10.
   You must define the following DD card names:
   CMPRT10 for the main task;

CMPRT101 to CMPRT1*nn* for all subtasks, *nn* = NTASKS-1;

Example:

```
//NAT31    JOB  CLASS = K,MSGCLASS = X
//NATSTEP  EXEC PGM=NATOS31,REGION = 8M
//STEPLIB  DD   DISP = SHR,DSN = SAG.NAT.LOAD
//         DD   DISP = SHR,DSN = SAG.ETB.LOAD
//CMPRMIN  DD   *
IM = D,MADIO = 0,MT = 0,OBJIN = R,AUTO = OFF,MAXCL = 0,ID=',',INTENS = 1,
PRINT = ((10),AM = STD)
/*
//SYSUDUMP DD   SYSOUT = X
//CMPRT10  DD   SYSOUT = X
//CMPRT101 DD   SYSOUT = X
//CMPRT102 DD   SYSOUT = X
//CMPRT103 DD   SYSOUT = X
//CMPRINT  DD   SYSOUT = X
//CMPRINT1 DD   SYSOUT = X
//CMPRINT2 DD   SYSOUT = X
//CMPRINT3 DD   SYSOUT = X
/*
```

Trace File Handling under CICS

Under CICS, Natural Advanced Facilities is required to write to the additional Report 10. If Natural Advanced Facilities is not installed or not wanted, the trace file is written to the Work File 10, provided that this file exists and Print File 10 has not been defined, otherwise the trace is disabled.

The Natural work file should be assigned to an extra-partitioned transient data queue.

Examples:

Natural definition:

```
NTWORK ((10),AM=CICS,DEST=RPCT,TYPE=TD)
```

CICS definition:

```
RPCTRAC DFHDCT TYPE=SDSCI,                      X
               BLKSIZE=136,                     X
               BUFNO=1,                         X
               DSCNAME=RPCTRACE,                X
               RECFORM=VARUNB,                  X
               RECSIZE=132,                     X
               TYPEFLE=OUTPUT
        SPACE
RPCT    DFHDCT TYPE=EXTRA,                      X
               DSCNAME=RPCTRACE,                X
               DESTID=RPCT,                     X
               OPEN=INITIAL
```

CICS Startup JCL:

```
RPCTRACE DD    SYSOUT=*
```

Trace File Handling in VSE/ESA Batch Mode

In VSE/ESA batch mode assign a trace file to the Printer Number 10.

Example:

```
// LIBDEF PHASE,SEARCH=(SAGLIB.NATnnn,SAGLIB.ETBnnn),TEMP
// ASSGN SYS000,READER  // ASSGN SYSLST,FEE  // ASSGN SYS050,FEF
// EXEC RPCVSEE3,RPCSIZE=AUTO,PARM='SYSRDR'
IM=D,MADIO=0,MT=0,OBJIN=R,AUTO=OFF,MAXCL=0,ID=',',INTENS=1,
PRINT=((10),AM=STD,SYSNR=50)
/*
```

Trace File Handling in BS2000/OSD Batch Mode

In BS2000/OSD batch mode assign a trace file to Printer Number 10.

Example:

```
/.RPCSERV         LOGON
/             SYSFILE   SYSOUT = output-file
/             SYSFILE   SYSDTA = (SYSCMD)
/             SYSFILE   SYSIPT = (SYSCMD)
/          FILE trace-file,LINK = P10,OPEN = EXTEND       */server trace file
/             STEP
/             SETSW      ON = 2
/             EXEC $DC1.NATDB64B
MADIO = O,IM = D,ID = ',',PRINT = ((10),AM = STD)
```

Handling Errors

- Remote Error Handling
- Avoiding Error Message NAT3009 from Server Program
- User Exit NATRPC01

Remote Error Handling

Any Natural error on the server side is returned to the client as follows:

- Natural RPC moves the appropriate error number to the *ERROR-NR system variable.
- Natural reacts as if the error had occurred locally.

Note:
If AUTORPC is set to ON and a subprogram cannot be found in the local environment, Natural will interpret this as a remote procedure call. It will then try to find this subprogram in the service directory.
If it is not found there, a NAT6972 error will be issued. As a consequence, no NAT0082 error will be issued if a subprogram cannot be found.

See also Using the RPCERR Program.

Avoiding Error Message NAT3009 from Server Program

If a server application program does not issue a database call during a longer period of time, the next database call might return a NAT3009 error message.

To avoid this problem, proceed as follows:

1. Add a FIND FIRST or HISTOGRAM statement in program NATRPC39, library SYSRPC.
2. Copy the updated program to library SYSTEM on FUSER.
   The STEPLIB concatenation of the library to which the server currently is logged on is not evaluated.

User Exit NATRPC01

This exit is called when a Natural error has occurred, actually after the error has been handled by the Natural RPC runtime and immediately before the response is sent back to the client. This means, the exit is called at the same logical point as an error transaction, that is, at the end of the Natural error handling, after all ON ERROR blocks have been processed.

In contrast to an error transaction, this exit is called with a CALLNAT statement and must therefore be a subprogram which must return to its caller.

The interface to this exit is similar to the interface of an error transaction. In addition, the exit can pass back up to 10 lines of information which will be traced by the Natural RPC runtime. Only lines which begin with a non-blank character will be traced.

Important Notes:

1. NATRPC01 must be located in library SYSTEM on FUSER. The STEPLIB concatenation of the library to which the server currently is logged on is **not** evaluated.
2. The DEFINE DATA PARAMETER block must not be changed.

Terminating a Natural RPC Server

- Using SYSRPC
- Using Entirex Control Center or EntireX System Management Hub
- User Exit NATRPC99

Using SYSRPC

Use the TE command of the Server Command Execution function.

A Natural RPC server can only be terminated if the server is currently neither executing a remote CALLNAT nor waiting for the next CALLNAT request in a conversation.

Using Entirex Control Center or EntireX System Management Hub

Use the Shutdown command (EntireX Control Center) or the Deregister Button (EntireX System Management Hub).

A Natural RPC server can only be terminated if the server is not currently executing a remote CALLNAT request.

User Exit NATRPC99

This exit is called after the Natural RPC server has deregistered and logged off from the server node.

- If no NATRPC99 is found, the server terminates immediately as usual.
- If NATRPC99 is found, the server continues to run as a normal Natural session.

NATRPC99 is called with a FETCH statement without any parameters, that is, no data is put on the Natural stack before NATRPC99 has been called.

You may add any coding to NATRPC99, including transfer control statements (FETCH, CALLNAT, PERFORM) and statements that terminate the program (STOP, ESCAPE, TERMINATE).

If NATRPC99 is terminated with a RETURN or STOP statement, Natural returns to the NEXT prompt. If the NEXT prompt is not supported in the environment used (CM=OFF, asynchronous Natural session, etc.) the session terminates. Otherwise, the session tries to read the next command from the primary input file/dataset for Natural commands and INPUT data (CMSYNIN).

Important Notes:

1. NATRPC99 must be a Natural program.
2. NATRPC99 is currently only called if the server is terminated with a TE command issued using the Server Command Execution function of the SYSRPC utility. The exit is not called if the server is terminated via the Entirex Control Center or the EntireX System Management Hub.
3. NATRPC99 must be located in the library SYSTEM on FUSER. The STEPLIB concatenation of the library to which the server currently is logged on is not evaluated to find NATRPC99.
4. Natural objects that are called by NATRPC99 (FETCH, CALLNAT, PERFORM) must be located either in the library to which the server is logged on or in one of its STEPLIBs (including SYSTEM FUSER).